

Optimizing failover time in a Serviceguard environment



Executive summary.....	2
The HP Serviceguard failover process.....	2
The process when failover is caused by a node failure	2
Node Failure Detection	3
Election of Cluster Membership	3
Lock Acquisition	4
Quiescence	4
Cluster Component Recovery	4
Standard Serviceguard implementation: Resource Recovery.....	5
Standard Serviceguard implementation: Applications Recovery.....	5
Serviceguard Extension for RAC: Group Membership Reconfiguration	5
Serviceguard Extension for RAC: RAC Reconfiguration	5
The process when failover is caused by a package failure	5
Standard Serviceguard implementation: Resource Failure Detection.....	6
Standard Serviceguard implementation: Package Determination	6
Standard Serviceguard implementation: Resource Recovery.....	7
Standard Serviceguard implementation: Application Startup.....	7
Serviceguard Extension for RAC: Group Membership Reconfiguration	7
Serviceguard Extension for RAC: RAC Reconfiguration and Database Recovery.....	7
How you can optimize failover time.....	7
Some help in estimating time for failover.....	8
Node timeout value	8
Testing	9
Lock acquisition (cluster lock, also called tie-breaker or arbitrator).....	10
Heartbeat subnet.....	10
Network failure detection	10
Number of nodes and number of packages	11
EMS resources	11
Package control scripts	11
Applications	12
Serviceguard Extension for Faster Failover	12
Requirements for SGeFF.....	13
Environments suitable to SGeFF	13
Cluster parameter considerations	14
Conclusion.....	15
For more information.....	15

Executive summary

One of the most important measurements for an effective high-availability/mission-critical environment is how much delay the end user notices in the event of a failure. In these environments, certain things need to take place, such as detecting a failure, finding a way to restart the work, ensuring data integrity, and restarting applications and making them available to users.

Different business needs require different environments. Environments vary widely in their tolerance for unplanned downtime, their hardware configuration, specialized software, and system and data management. These factors require careful consideration when configuring a high-availability environment. Thorough testing in a production or near-production environment should be done to make sure that the configured cluster meets the requirements. Testing and fine-tuning can help optimize failover time and increase application availability to end users.

This paper explains the HP Serviceguard failover process. Then it discusses how you can optimize your cluster failover time. Then it discusses Serviceguard Extension for Faster Failover, a Serviceguard auxiliary product you can purchase that reduces the time for the Serviceguard component of the failover process.

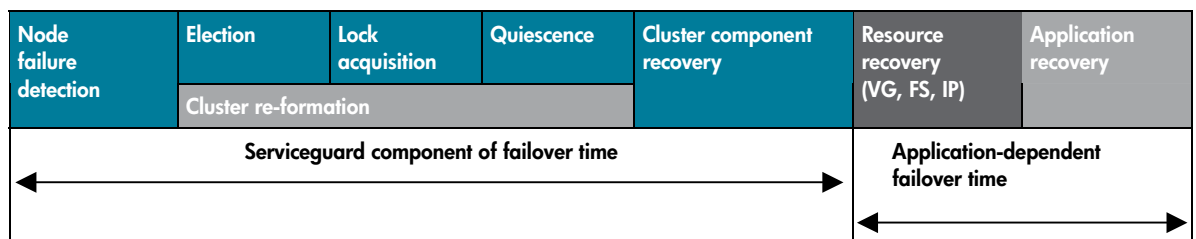
The HP Serviceguard failover process

The process when failover is caused by a node failure

Serviceguard nodes monitor each other to be sure they can all communicate and cooperate. Every node in a Serviceguard cluster sends heartbeat messages over the network and listens for heartbeat messages from other nodes. Heartbeat messages are sent at regular intervals, defined in the cluster configuration file as the HEARTBEAT_INTERVAL.

If a node does not receive a heartbeat from another node, it begins the process of re-forming the cluster and removing the unreachable node from cluster membership. Figure 1 shows the steps in a failover caused by a failed node.

Figure 1. Steps in a failover caused by a failed node—standard Serviceguard implementation



Note: Diagram is not to scale.

The Serviceguard component of the total failover time when it is caused by node failure (not a package failure) is composed of: node failure detection, election, lock acquisition, quiescence, and cluster component recovery.

- **Node failure detection**—The system notices that a cluster node is not in communication with the other cluster nodes. Serviceguard begins to re-form the cluster.
- **Election**—The cluster nodes decide which nodes will be in the re-formed cluster.
- **Lock acquisition**—If more than one group of nodes wants to re-form the cluster and no group has a clear majority of members, the first group to reach the cluster lock re-forms the cluster.

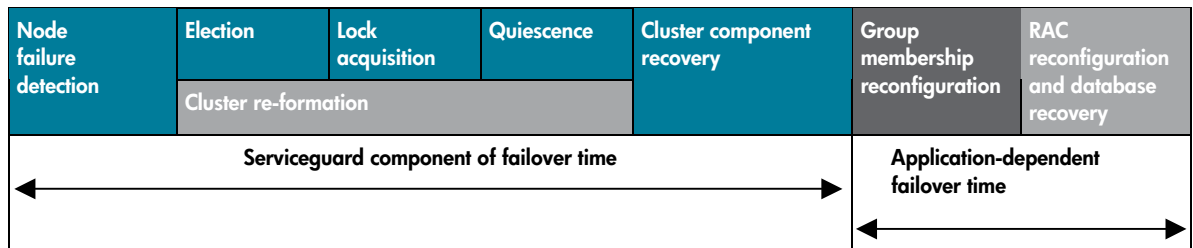
- **Quiescence**—During this quiet waiting time, non-members of the newly formed cluster are rebooted.
- **Cluster component recovery**—Serviceguard does miscellaneous tasks, such as cluster information synchronization and package determination, before the cluster resumes the work.

During the application-dependent phase of the failover time, Serviceguard starts the package control scripts, which were written by the user. In standard Serviceguard implementations, there are two steps, as shown in Figure 1.

- **Resource recovery**—The package’s resources are made available.
- **Application recovery**—If applications or processes were moved to a new node, they are restarted.

As shown in Figure 2, the application-dependent steps are a little different for an Oracle® Real Application Cluster (RAC) package in a cluster with Serviceguard Extension for RAC.

Figure 2. Steps in a failover caused by a failed node—Serviceguard Extension for RAC implementation



Note: Diagram is not to scale.

The two application-dependent steps for a RAC implementation are:

- **Group membership reconfiguration**—If there is a change in membership, RAC starts the reconfiguration.
- **RAC reconfiguration and database recovery**—After a cluster membership change, RAC reassigns the database locks that were on failed nodes and restarts the databases.

Node Failure Detection

If a node does not get a heartbeat message from another node, it will declare the other node unreachable. A node may be unreachable for many reasons. There may be a transient interruption that can recover automatically in a short time, such as a spike in network activity, I/O, or CPU, or a temporary kernel hang. Or there may be a failure that will not recover automatically or quickly, such as a hardware or power supply failure, or a crashed operating system.

A node is considered failed if there is no heartbeat during time specified as `NODE_TIMEOUT` in the cluster configuration file. After the `NODE_TIMEOUT` value is reached, Serviceguard begins to re-form the cluster without the failed node.

Election of Cluster Membership

After a node believes another node has failed, it begins the cluster re-formation process. The cluster nodes elect the nodes that will be members of a newly re-formed cluster.

Each healthy node tries to take over the work of the cluster. It tries to change cluster membership to include the nodes it can communicate with and exclude the nodes it cannot reach.

If one node has failed but all the others can still communicate with each other, the others quickly form a group that excludes that node. However, it could be that a group of healthy nodes cannot communicate with some other healthy nodes. In this case, several groups could try to form a cluster.

The group that achieves quorum will become the new cluster. There are two ways for a group to achieve quorum:

- If the group includes more than half of the nodes that were active the last time the cluster was formed, it has quorum because it has the majority.
- If two groups each have exactly half of the nodes that were active the last time the cluster was formed, the group that acquires the cluster lock achieves quorum. (See “Lock acquisition” for more about the cluster lock.)

The group that achieves quorum takes over the work of the cluster. The excluded nodes are not allowed to proceed in cluster re-formation, and they will be rebooted.

The amount of time taken for election depends primarily on the values for the `NODE_TIMEOUT` and `HEARTBEAT_INTERVAL`. If there are a number of temporary interruptions and recoveries, nodes may lose contact and then resume contact several times. If so, the process will take longer because of repeated elections.

Lock Acquisition

If two equal-sized groups try to re-form the cluster, the cluster lock acts as arbitrator or tie-breaker. Whichever group acquires the cluster lock will achieve quorum and form the new cluster membership.

Serviceguard uses three types of cluster locks:

- Quorum server (HP-UX and Linux[®])
- Lock disk (HP-UX only)
- Lock disk LUN (Linux only)

A two-node cluster is required to have a cluster lock. In clusters of three or more nodes, a lock is strongly recommended. The lock disks can be used for clusters with two, three, or four nodes. A quorum server can be used on a cluster of any size.

Acquiring a quorum server or a lock disk LUN usually takes less time than acquiring a lock disk.

Quiescence

Quiescence is a quiet waiting time after new cluster membership is determined. Nodes that are not in the new membership are forcibly rebooted. The waiting time is a protection against data corruption. Its purpose is to make sure that the reboot finishes so an excluded node is not trying to run a package or issue any I/O.

Quiescence is important when some nodes in the cluster cannot communicate with the others but could still run applications, particularly if the nodes have access to a common database.

Quiescence is calculated by Serviceguard, and the user cannot directly change it.

Cluster Component Recovery

In this short step (typically less than one second), Serviceguard does miscellaneous tasks, such as cluster information synchronization and package determination. If packages are down because of a failure, Serviceguard determines whether it should restart the package and which node (or nodes) it should start them on. (See “Standard Serviceguard implementation: Package Determination” on page 6.)

The time needed for cluster component recovery depends mainly on how many packages need to be restarted. At the end of this recovery phase, Serviceguard starts the packages by starting the packages’ control scripts one after the other, using an “exec” process.

The user cannot directly change the time needed for cluster component recovery.

Standard Serviceguard implementation: Resource Recovery

When Serviceguard starts a package's control script, the application-dependent part of failover begins. Package resources are made available, ready for the package's applications to start. Package resources include IP addresses, file systems, volume groups, and disk groups needed by the package. Some resources may require other recovery steps before they can be used.

The time to complete resource recovery is determined largely by the package control script and depends on the package's applications, services, and resources.

Standard Serviceguard implementation: Applications Recovery

The control scripts' commands complete the application-dependent part of failover, recovering and restarting package applications. The amount of time it takes depends on the applications and how they are configured.

Serviceguard Extension for RAC: Group Membership Reconfiguration

When Serviceguard Extension for RAC communicates the group membership to Oracle RAC, the application-dependent part of a RAC failover starts. If there is a change in membership, RAC will start reconfiguration. RAC needs to know which nodes are in the re-formed cluster; if the node holding the database lock leaves the cluster, another node needs to claim the lock.

The time needed for group membership reconfiguration is determined by RAC, and the user cannot directly change it.

Serviceguard Extension for RAC: RAC Reconfiguration

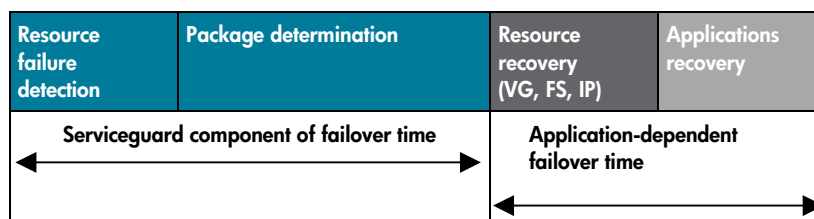
After Oracle RAC is notified of a cluster membership change, it starts its own reconfiguration to claim the database locks that were on failed nodes. RAC reconfiguration and recovery occurs in the RAC instances running on the other nodes in the cluster.

The time needed for this step is determined by RAC, and the user cannot directly change it.

The process when failover is caused by a package failure

These are the steps in a failover that is caused by a package failure (rather than a node failure).

Figure 3. Steps in a failover caused by package failure—standard Serviceguard implementation



Note: Diagram is not to scale.

The steps in the Serviceguard component of failover are:

- **Resource failure detection**—Serviceguard notices that a monitored service or resource is down.
- **Package determination**—Serviceguard decides whether, and where, to restart the packages.

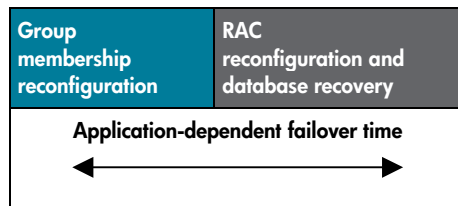
With standard Serviceguard implementations, the steps in the application-dependent component are:

- **Resource recovery**—Using the Package Control Scripts, Serviceguard makes the resources available to the packages.
- **Application recovery**—This time is to restart applications or processes that were moved to a new node.

With Serviceguard Extension for RAC, failure of a RAC package does not trigger any failover action from Serviceguard. If there is a failure, such as a database instance crash, the Serviceguard cluster will keep running without re-forming, so there is no Serviceguard component to the failover.

Oracle RAC re-forms a new membership and performs database recovery.

Figure 4. Steps in a failover caused by package failure—Serviceguard Extension for RAC implementation



Note: Diagram is not to scale.

With RAC, the two application-dependent steps of failover are different from the steps with standard Serviceguard implementations. They are:

- **Group membership reconfiguration**—If there is a change in membership, RAC will start reconfiguration.
- **RAC reconfiguration and database recovery**—After a cluster membership change, RAC reassigns the database locks that were on failed nodes and restarts the databases.

Standard Serviceguard implementation: Resource Failure Detection

Serviceguard monitors the configured package services and networks.

Serviceguard package configuration can include Event Monitoring Service (EMS), which monitors hardware such as storage. EMS polls an EMS resource monitor to get returns. The polling interval is set in the package configuration file. When EMS finds a resource failure, it immediately notifies Serviceguard.

Generally, when a resource fails, the package will fail over to another node. If the package is configured with `NODE_FAIL_FAST_ENABLED` set to “yes”, Serviceguard will cause the node to fail. If this happens, the process will start at the first step described in the previous section, “The process when failover is caused by a node failure.”

Standard Serviceguard implementation: Package Determination

When a package fails, Serviceguard can automatically try to restart it if `AUTO_RUN` is set to YES in the package configuration file. If `AUTO_RUN` is set to YES, Serviceguard next determines where to start the package. It creates an ordered list of nodes, which is prioritized according to the node list and the settings of the failover and failback policies in the package’s configuration file.

The user cannot directly change the time needed for package determination.

Standard Serviceguard implementation: Resource Recovery

Serviceguard starts each package's control script to begin the application-dependent part of failover. During failover, package resources need to be made available before applications can be started. The package resources include IP addresses, file systems, volume groups, and disk groups. Some resources may require recovery before they can be used. Instructions for this are in each package's control script.

The time required for resource recovery depends on the number of resources and the instructions in the control script.

Standard Serviceguard implementation: Application Startup

The control scripts complete the application-dependent part of failover by following commands for the recovery and restart of package applications. Some packages and some applications may also do their own recovery before startup.

The time required for application startup depends on each application and how it is configured.

Serviceguard Extension for RAC: Group Membership Reconfiguration

RAC group membership reconfiguration is the same whether failover is caused by a node failure or a package failure. To start group membership reconfiguration, Serviceguard Extension for RAC communicates the group membership to Oracle RAC. If there is a change in membership, RAC will do the reconfiguration.

The time required for group membership reconfiguration depends on the RAC configuration.

Serviceguard Extension for RAC: RAC Reconfiguration and Database Recovery

After Oracle RAC is notified of a cluster membership change, it starts its own reconfiguration and recovery.

The time required for RAC reconfiguration and database recovery depends on the RAC configuration.

How you can optimize failover time

There are ways you can optimize the failover process for your environment to reduce the time a package is unavailable. If the failover process takes longer than necessary, you are not getting maximum availability. If the failover starts and completes too quickly, however, you may get unnecessary failovers that reduce performance of a cluster—possibly reducing availability instead of improving it. It is important to find a balance between the extremes.

The optimal failover time would be long enough to allow for recoverable interruptions, but no longer than that.

The time required for the Serviceguard portion of failover depends largely on the node timeout value, but it also depends on the heartbeat interval, the number of nodes in the cluster, and the cluster lock used. There are ways to fine-tune these factors to help optimize failover.

To set the Serviceguard parameters, you need to determine the likelihood of transient interruptions and the amount of time it takes them to recover and continue. If your cluster is in a busy environment, you need to tolerate interruptions or you will get unnecessary—and possibly repeated—failovers. If, however, your networks and systems do not get overloaded, you can set your failover parameters more aggressively.

Try to tune the environment so there are fewer interruptions and it takes less time to recover from them. Consider ways to distribute the workload across the cluster. Consider adding a node to the cluster. Because your timeout value has to allow for the longest recovery times, you can reduce the timeout value if you can smooth out the peaks.

Consider the time it takes for the application-dependent component of failover. If your applications need a short time for recovery and restart, you can afford to set your failover parameters more aggressively. If your application recovery and restart is quick, it is an advantage to have a quick reaction to failures. However, if it takes a long time for your applications to restart or for your databases to recover, set the timeout value more conservatively. Before you start a lengthy failover, it is an advantage to wait a bit for a transient problem to recover on its own.

Some help in estimating time for failover

The following table can help you estimate the total failover time for your Serviceguard cluster.

Failover component name	Time estimate
Resource failure detection	
Network failure detection	6 x NETWORK_POLLING_INTERVAL
EMS resource failure detection	RESOURCE_POLLING_INTERVAL
Service failure detection	Immediately
Node failure detection	NODE_TIMEOUT
Cluster membership re-formation time (Serviceguard component of failover time in case of node failure)	<p>This depends mostly on NODE_TIMEOUT. It is also affected by the number of heartbeat networks, the type of cluster lock, and HEARTBEAT_INTERVAL.</p> <p>For example, the re-formation time is 28 seconds if NODE_TIMEOUT is set to 2 seconds, there is more than one heartbeat subnet, and a quorum server is configured with QS_TIMEOUT_EXTENSION set to 0.</p> <p>On HP-UX, check the time with the <i>cmquerycl</i> command: After configuring a cluster, issue the command <i>cmquerycl -c <cluster_name></i> and observe the output. The value next to the cluster lock is the cluster membership re-formation time.</p>
Cluster component recovery	This depends on the number of EMS resources, packages, nodes, etc. This time is usually very small compared to total failover time.
Resource recovery	This depends on the number of volume groups, IP addresses, services, etc. It usually ranges from a low of less than 1 second to a high of several minutes.
Application startup and recovery time	This is totally dependent on the application.

Node timeout value

To help optimize failover time, first consider fine-tuning the setting for NODE_TIMEOUT in your cluster configuration file. Changing this probably will make the greatest difference in your cluster failover time. When a node times out, Serviceguard declares the node failed and begins cluster re-formation.

For Serviceguard, the range of supported values of NODE_TIMEOUT is 2 to 30 seconds. The recommended value is from 5 to 8 seconds for most clusters.

Reducing the node timeout decreases the time to detect node failures, which can decrease the total failover time. However, a small node timeout value also introduces a risk. If there are temporary interruptions and you set the timeout value so low that the node cannot recover communication, the node might fail unnecessarily or the cluster might re-form unnecessarily.

Setting the parameters too low may cause failovers that you could avoid. If you set your parameters so low that re-formation can complete before an unreachable node can recover from a temporary interruption, the node will be forcibly rebooted. Any packages running on it will be failed-over to another node.

If the unreachable node can recover communication before the re-formation completes, the node will rejoin the cluster. If you see that re-formation has taken place but nothing has changed, your timeout has already approached its limit; do not try to shorten the overall failover time. Another indication that you have approached the limit is a message in syslog that says, "Warning: cmcld process was unable to run for the last <xxx> seconds."

If your application takes a long time to recover and restart, set the node timeout conservatively. If your database takes several minutes to recover, it isn't worth risking an unnecessary failover to shave a few seconds off of the Serviceguard failover time.

If your application restarts quickly, you can afford to set the node timeout more aggressively. When your application only takes a few seconds to restart, there is little benefit in waiting a few seconds for an interruption to recover. You can afford to try a short timeout when you have short recovery and restart times.

Small, lightly loaded systems are likely to have fewer interruptions, and they are likely to recover more quickly. Highly loaded systems with a large number of disks are likely to have more frequent interruptions, and they are likely to take longer to recover. Try to spread the load and avoid spikes in activity. Set the node timeout to allow recovery time for interruptions when the load is heaviest.

Virtual partitions may have different latency characteristics than independent nodes due to hardware and firmware sharing. If you will be using virtual partitions in your cluster, there may be additional considerations when testing the node timeout value for your configuration. For more information, see the white paper "Serviceguard Cluster Configuration for Partitioned Systems", available from www.docs.hp.com/hpux/ha.

Testing

To fine-tune the parameters, it is important to test the cluster in an environment that imitates the actual production environment. Test the cluster, running all of its packages, with the heaviest expected loads on networks, CPU, and I/O.

To time failover, force each package to fail-over to another node. One way to force a failover is to power off the node where the package is running. Read the logs of the failover, noting the time stamps.

Change parameters in small increments, then re-evaluate and re-test. Check the system log. If there are indications of interruptions or transient problems, try to determine the recovery time. Look for re-formations that occurred without a change in cluster membership, which indicate you have reached the lower time limit.

Try different settings for the node timeout until you find the optimal one. You want a value that results in the shortest failover time without any unnecessary re-formations or failovers from recoverable temporary problems.

Allow a margin of safety for the tested node timeout value. How much time to allow depends on how closely your test environment reflects your actual environment at its busiest.

Re-test and re-evaluate your settings periodically, especially when new disks, new networks, or new applications are added to the cluster. Monitor traffic on heartbeat networks; watch for increases in traffic, especially ones that could cause temporary spikes.

Lock acquisition (cluster lock, also called tie-breaker or arbitrator)

Your choice of lock disk may help you optimize failover time. If you set the node timeout to less than 6 seconds, a quorum server, lock LUN, or Logical Volume Manager (LVM) SCSI lock disk will shorten the time for lock acquisition. If the node timeout is more than 6 seconds, however, the type of lock will probably not affect the total failover time.

Different lock disks have different lock acquisition times. If you use a lock disk or plan to use one, consult its documentation to see how long acquisition takes.

- It takes about 10 seconds to acquire an LVM SCSI lock disk.
- It takes about 32 seconds to acquire a Fibre Channel LVM lock disk.
- The time to acquire a lock LUN SCSI or a quorum server increases with the value of `NODE_TIMEOUT`, but it is not directly proportional. For example, if the node timeout is 2 seconds, acquisition takes about 8 seconds; when the node timeout is set to 4 seconds, it takes about 14 seconds.

If you choose a quorum server, be sure the network between it and the nodes is highly available and reliable. If you have delays reaching the quorum server, you can configure a `QS_TIMEOUT_EXTENSION`, but the extension time adds directly to the lock acquisition time.

Serviceguard calculates the time for the actual lock acquisition. You cannot change it directly, but you can configure a faster lock device to reduce the lock acquisition time.

Heartbeat subnet

If you have one heartbeat configured, with the required standby LAN, and if `NODE_TIMEOUT` value is less than 4 seconds, you can reduce failover time if you configure multiple heartbeats instead. Since heartbeat messages are sent over all heartbeat subnets concurrently, there will be no wait for network switching if a primary LAN fails. To avoid delays from busy networks, configure at least one private dedicated network for heartbeat.

Keep in mind that certain configurations, such as those using VERITAS Cluster Volume Manager (CVM), do not allow multiple heartbeats.

Network failure detection

The `NETWORK_POLLING_INTERVAL` specifies how often Serviceguard checks its configured networks. In general, the default works best.

Frequent polling allows Serviceguard to respond more quickly to LAN failure. A quick response could reduce the package failover time simply because failover starts earlier. However, if the polling interval is too short, the frequent traffic could just make the network and the system busier.

Number of nodes and number of packages

Serviceguard's Package Manager starts the package control scripts using the exec process. This time is noticeable only when you have a large number of packages configured on a small number of nodes. On most systems, for example, if 50 packages failed over to one node, it would take a few seconds to start the scripts.

Slight effects are seen in two other places:

- During re-formation, clusters of more than eight nodes might need extra time to synchronize cluster information.
- During resource recovery, the Package Manager has two tasks. First, it checks the packages to determine which ones failed. The more packages there are, the more time this could take. Then it needs to determine which nodes should adopt the packages and run them after re-formation. The more packages each node has, the more time this could take.

EMS resources

There are two factors to consider about your EMS resources:

- **EMS resource monitor detection time**—This depends entirely on the EMS resource monitor and how it works. Look at the monitor's documentation; usually you can set this time.
- **The time for the EMS message to get to Serviceguard**—At most, this takes as long as the time set for RESOURCE_POLLING_INTERVAL. In the package configuration file, you want to set the interval low enough to discover failure quickly. However, if you set it too low, frequent polling just makes the network and the system busier.

Package control scripts

When many storage units are involved, you might be able to reduce resource recovery time to help optimize failover. Refer to the chapter "Optimizing for Large Numbers of Storage Units" in the *Managing Serviceguard* manual for your version of Serviceguard. Manuals are available from www.docs.hp.com/hpux/ha -> Serviceguard.

The type of file system can greatly reduce the time it takes for file consistency checks. For packages on HP-UX, VxFS is faster than HFS. On Linux, a journaled file system (such as Reiser FS and ext3 FS) is faster than a non-journaled file system (ext2).

Adding or removing IP addresses takes some time and affects failover time. On HP-UX, it takes at least one second longer to add an IPv6 address if you enable duplicate address detection (DAD). For more information, see "IPv6 Relocatable Address and Duplicate Address Detection" in the *Managing Serviceguard* manual for your version for Serviceguard, available from www.docs.hp.com/hpux/ha -> Serviceguard.

Make your control scripts as efficient as possible. The time needed to start and stop services adds to the total failover time. Streamline any customer-defined functions to help save time.

Applications

There is no single solution to optimizing the efficiency of the many different applications. However, here are some general tips.

Failed-over applications may spend a long time recovering data. See if you can reduce this time. Consider contacting the application vendor and the systems integrator for specific tuning tips.

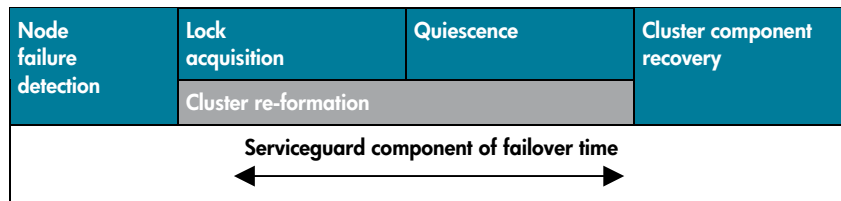
If a database management system is used, consider implementing Oracle RAC. RAC significantly reduces resource recovery after the failure of a RAC instance, so it helps reduce application recovery time. To use Oracle RAC in a Serviceguard environment, purchase Serviceguard Extension for RAC.

Serviceguard Extension for Faster Failover

Further optimization of Serviceguard failover time is possible with Serviceguard Extension for Faster Failover (SGeFF). This auxiliary product for Serviceguard on HP-UX is made for customers who need faster failover and can configure a specific type of cluster.

The Serviceguard component of failure changes when Serviceguard Extension for Faster Failover is enabled, as shown in Figure 5.

Figure 5: Steps in a failover with Serviceguard Extension for Faster Failover



Note: Diagram is not to scale.

SGeFF reduces the Serviceguard component of failover time by:

- Eliminating the time ordinarily reserved for election
- Shortening the time for cluster lock acquisition
- Eliminating the time ordinarily required for local switching (see “Heartbeat Subnet” on page 10)

SGeFF cannot change the failover time of application-dependent components such as databases and applications.

Requirements for SGeFF

Every SGeFF cluster must meet these configuration requirements:

- Serviceguard A.11.16 and SGeFF must be installed on every node in the cluster.
- The cluster must have no more than two nodes.*
- The cluster must have a quorum server cluster lock installed outside the cluster.
- More than one heartbeat network must be configured.
- An RS-232 link is not allowed for a heartbeat network.
- CVM cannot be used in an SGeFF cluster because CVM requires a single heartbeat and SGeFF requires multiple heartbeats.

If Fibre Channel storage is used with SGeFF, it is recommended that one or two switches be configured between nodes and storage. This reduces the time I/O might spend in the fabric, which becomes critical if a node takes over an application from a failed node that was accessing a Fibre Channel disk.

An SGeFF cluster cannot expand online to a non-SGeFF cluster of more than two nodes. SGeFF can only be enabled or disabled when the cluster is halted.

If SGeFF is used with extended-distance clusters, use the “two data centers and third location” configuration described in “Designing Disaster Tolerant High Availability Clusters”, available from www.docs.hp.com/hpux/ha -> Metrocluster or Continentalcluster.

Environments suitable to SGeFF

Systems that have moderate and predictable loads are candidates for SGeFF. Suitable systems should be free from frequent or large spikes in CPU, network, or I/O activity that would affect the normal operation of Serviceguard.

It is important to remember that SGeFF reduces only the Serviceguard component of failover time; application-dependent failover time is unaffected. As mentioned previously in “How you can optimize failover time”, it is important to reduce both the Serviceguard component of failover time and the application-dependent failover time. If it takes a long time (say 5 minutes) to start up the application and complete recovery, use of SGeFF does not reduce total failover time significantly.

SGeFF is well-suited for:

- Applications that have no or very low startup time
- Applications that have very rapid recovery time
- Applications that have fewer and/or quick recoverable resources (for example, packages with fewer volume groups, fewer file systems, VxFS for faster recovery)

Here is an example of an environment well-suited for SGeFF:

- An SGeRAC environment where
 - The Oracle RAC instances are already running on all of the nodes, so there is no application startup time
 - The Oracle RAC can be tuned for rapid recovery, often in 10–60 seconds
- An application with conventional database software tuned to minimize recovery time
- Any IP-based application with low recovery time

* SGeFF clusters must have two nodes. Configuring a one-node cluster is allowed and may be useful to begin configuration; however, no failover is possible with only one node.

Cluster parameter considerations

There are risks associated with Serviceguard Extension for Faster Failover's shorter failover time. If there is a temporary interruption, the timeout should be long enough for the interruption to recover. Each administrator must decide how much time the cluster needs to confirm the likelihood (if not certainty) of a failure. On a busy system where the networks, I/O, or CPU have frequent or large spikes in activity, transient problems are likely to cause delayed heartbeats.

With SGeFF, a minimum value of 1.6 seconds for `NODE_TIMEOUT`, with 0.8 seconds for `HEARTBEAT_INTERVAL`, is supported. With these values, the Serviceguard component of failover time can be reduced to 5 seconds. Such low values are not suitable for all SGeFF environments.

`NODE_TIMEOUT` must be carefully set and carefully tested in an SGeFF cluster. The Faster Failover cluster re-formation process finishes immediately, so it becomes more likely that a healthy node with transient problems will be timed out of the cluster.

Node timeout should not be so short that it cannot tolerate transient problems or temporary interruptions. Each administrator must determine the optimal time that Serviceguard should wait for interruptions to recover before it times out and acts.

Node timeout should not be so short that a delayed heartbeat from a healthy node causes the cluster to begin re-forming. If it is, a node may be taken out of the cluster unnecessarily, or the node may recover in time to rejoin the cluster. In the second case, the cluster will re-form, but the membership will be just the same as before the re-formation.

The value of `NODE_TIMEOUT` will have a large effect in a Faster Failover cluster. For example, two identical two-node clusters have a configuration valid for Faster Failover. One has a standard Serviceguard implementation, the other has SGeFF enabled. They both have `QS_TIMEOUT_EXTENSION` set to zero and `NODE_TIMEOUT` set to 2 seconds. One has a standard Serviceguard implementation; one has SGeFF enabled. If there is a transient problem that lasts for 8 seconds, the two clusters will have different results:

- With a standard Serviceguard implementation, re-formation will take about 28 seconds. The transient problem will recover before re-formation is done, so the node will stay up and will be able to rejoin the cluster.
- With SGeFF, the re-formation will complete in 6 seconds, far less than the time needed for the problem to recover. The cluster will re-form without it. The node will be rebooted.

Failover in an SGeFF cluster is much faster than failover in a standard Serviceguard cluster. Using the example SGeFF cluster discussed above, an SGeFF cluster can be set to twice the node timeout and still require less than half the time for the Serviceguard component of failover:

- With a standard Serviceguard cluster installed as discussed above, the timeout value of 2 seconds completed the Serviceguard component of failover in 28 seconds.
- With SGeFF installed and the timeout value changed to 4 seconds, the Serviceguard component of failover will complete in 12 seconds.

Virtual partitions may have different latency characteristics than independent nodes due to hardware and firmware sharing. If you will be using virtual partitions in your cluster, there may be additional considerations when testing the node timeout value for your configuration. For more information, see the white paper "Serviceguard Cluster Configuration for Partitioned Systems", available from www.docs.hp.com/hpux/ha.

Conclusion

The total failover time in a Serviceguard cluster depends on many different factors and the interactions between them. Serviceguard users can optimize failover time to help maximize the time applications are available.

Optimizing failover time means finding a balance between waiting longer than needed to act and acting too hastily. Less time spent in the failover process means less time packages are unavailable. If a cluster detects failures quickly, it can re-form and restart its applications quickly. But if the timing is set too aggressively, it can result in unnecessary—and possibly repeated—failovers, ultimately decreasing overall availability.

The total time for failover includes a Serviceguard component and an application-dependent component. Over-emphasis on optimizing only one component can result in higher unavailability.

To help reduce the time for the Serviceguard component, examine the settings for several timing parameters, primarily NODE_TIMEOUT.

To reduce the time for application startup and recovery, examine the control script and each application's documentation.

Serviceguard Extension for Faster Failover, a product purchased separately, can further reduce Serviceguard's failover time to as little as 5 seconds. It is only supported with certain configurations.

For more information

To learn more about HP high-availability products and solutions, please visit:
www.hp.com/go/serviceguard

© 2004 Hewlett-Packard Development Company, L.P. The information contained herein is subject to change without notice. The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein.

Linux is a U.S. registered trademark of Linus Torvalds. Oracle is a registered trademark of Oracle Corporation, Redwood City, California.

5982-6793EN, 06/2004

